CS579 Assignment 1

Aakef Waris

September 8, 2020

Problem 1

1a)

Tracing through djikstra's algorithm
Source Node = v1
Unvisited = [v1, v2, v3, v4, v5, v6, v7]
Distance from source = [0, ∞, ∞, ∞, ∞, ∞, ∞]

Current Node = v1
Updated Distances = [0, 7, ∞, 5, ∞, ∞, ∞]
Updated Visited = [v2, v3, v4, v5, v6, v7]
Min Distance to unvisited = 5
New Node = v4

Current Node = v4
Updated Distances = [0, 7, ∞, 5, 20, 11, ∞]
Updated Visited = [v2, v3, v5, v6, v7]
Min Distance to unvisited = 7
New Node = v2

Current Node = v2
Updated Distances = [0, 7, 15, 5, 14, 11, 22]
Updated Visited = [v3, v5, v6, v7]
Min Distance to unvisited = 11
New Node = v6

Current Node = v6
Updated Distances = [0, 7, 15, 5, 14, 11, 22]
Updated Visited = [v3, v5, v7]
Min Distance to unvisited = 14
New Node = v5

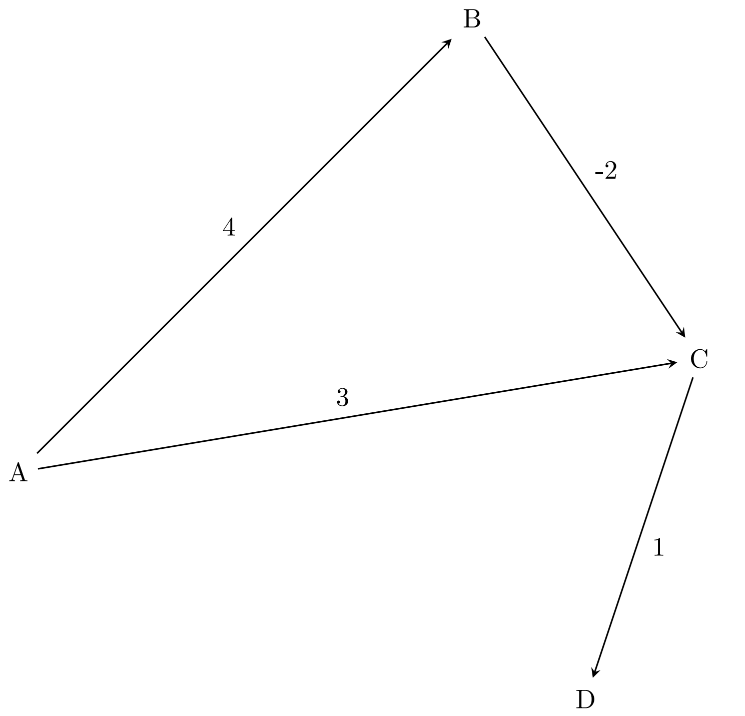
Current Node = v5
Updated Distances = [0, 7, 15, 5, 14, 11, 22]
Updated Visited = [v3, v7]
Min Distance to unvisited = 15
New Node = v3

Current Node = v3
Updated Distances = [0, 7, 15, 5, 14, 11, 22]
Updated Visited = [v7]
Min Distance to unvisited = 22
New Node = v7

Current Node = v7
Updated Distances = [0, 7, 15, 5, 14, 11, 22]
Updated Visited = []
Min Distance to unvisited = 22
New Node = None

Node	Distance from v1
v2	7
v3	15
v4	5
v5	14
v6	11
v7	22

1b)



Source Node: A

1c)

Djikstra's algorithim, will not always produce the shortest graph for a graph that has negative weights, even if it doesnt have a negative cycle. The primary issue, is that once a node is visited, it is no longer visted again. It is possible a node *V* exists, that acts as a bridge node to another subgraph. To get to this node from source there could be 2 paths:

Path A: A single edge path with an arbitrary weight x
Path B: A multi edge path, where the first edge has a weight y, and y > x, but the sum of all edge weights from the source node to node v, is less than the weight x, due to negative weights.

Djikstra's algorithm, will initially take Path B to node V, and use that distance to compute all distances to the nodes of the subgraph which is bridged by Node V. However later on the algorithm, when a shorter path is computed for Node V it will not be revisited, becuase it has already been revisited to compute nodes in the subgraph. So those weights will not update.

Problem 2

2a)

In a real social network, it is more reliable to use BFS over DFS, becuase you are able to get the immediate neighbors of a given node, which can lead to valuable information(e.g. Community detection). DFS, will continue to traverse a single path until a node is reached, where it has no neighbors. This doesnt provide much value, and could take a while, unless it was halted at an arbitrary point.

Problem 3

3a)

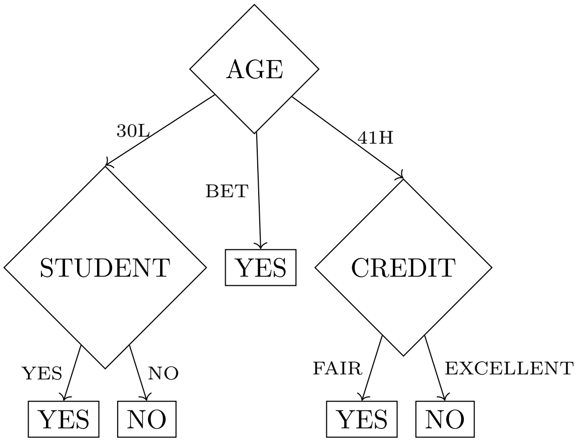
- Age: Ordinal
- Income: Nominal
- Student: Nominal

- Credit Rating: Nominal

3b)

When Age is discretized, it is now Nominal.

3c)



Listing 1: Insert code directly in your document

```
import math
def entropy(pi):
    total = 0
    if sum(pi) == 0:
        return 0
    for p in pi:
        p = p / sum(pi)
        if p != 0:
            total += p * math.log(p, 2)
        else:
            total += 0
    total *= -1
    return total

def gain(a):
    '''
    return the information gain:
    '''
    d = [sum(lst) for lst in a]
    if sum(d) == 0:
        return 0
    total = 0
    for v in a:
        total += sum(v) / sum(d) * entropy(v)

    gain = entropy(d) - total
    return gain
```

Entropy of the dataset: 0.94, with 9:5 ratio of Bought : Didn't Buy

Feature that brought the highest gain was AGE − > 0.88

3 branches: 30L, BET, 41H
In the 30L branch, the feature that brought the highest gain was Student− >0.97
This led to a perfect split, Students bought computers, and everyone else didn't

In the BET branch, everyone bought computers

In the 41H branch, the feature that brought the highest gain was Credit− >0.97
This led to a perfect split, Fair Credit bought computers, and Excellent didn't

Problem 4

P(Buy PC = YES) = 9/14
P(Buy PC = NO) = 5/14

P(AGE = 30L |Buy PC = YES) = 2/9
P(AGE = 30L |Buy PC = NO) = 3/5

P(INCOME = Low |Buy PC = YES) = 2/9
P(INCOME = Low |Buy PC = NO) = 1/5

P(STUDENT = True |Buy PC = YES) = 6/9
P(STUDENT = True |Buy PC = NO) = 1/5

P(CREDIT = Fair |Buy PC = YES) = 6/9
P(CREDIT = False |Buy PC = NO) = 2/5

P(Instance 15|Buy PC = YES) = 2/9 * 2/9 * 6/9 * 9/14 = 0.014

P(Instance 15|Buy PC = NO) = 3/5 * 1/5 * 1/5 * 2/5 * 5/14 = 0.003

Naive Bayes would classify instance 15 as Buy PC = TRUE